1.a) Mapa de Karnaugh e simplificação da função F(A,B,C,D)

A função dada é:

$F(A,B,C,D)=\sum m(2,3,5,10,14)+\sum md(0,4,8,12)$

Passos:

1. **Construir o mapa de Karnaugh:**

Primeiro, cria-se o mapa de Karnaugh de 4 variáveis (A, B, C, D). Como A é a variável de maior peso, seguimos a ordem de A, B nas linhas e C, D nas colunas.

Mapa de Karnaugh (4 variáveis):

AB\CD	**00**	**01**	**11**	**10**
00	m(0)	m(1)	m(3)	m(2)
01	m(4)	m(5)	m(7)	m(6)
11	m(12)	m(13)	m(15)	m(14)
10	m(8)	m(9)	m(11)	m(10)

Coloca-se a informação fornecida:

- **Mintermos (1):** 2, 3, 5, 10, 14
- **Indiferenças (X):** 0, 4, 8, 12
- **Restante (0):** Todos os outros valores

O mapa fica:

AB\CD	**00**	**01**	**11**	**10**
00	X	0	1	1
01	X	1	0	0
11	X	0	0	1
10	X	0	0	1

2. **Simplificação (Soma de Produtos):**

A partir do mapa de Karnaugh, identificam-se os laços (grupos de 1s) para simplificar a expressão:

AB\CD	**00**	**01**	**11**	**10**
00	X	0	1	1
01	X	1	0	0
11	X	0	0	1
10	X	0	0	1

- **Grupo de 1s em m(2) e m(3):** /A /B C
- **Grupo de 1s em m(10) e m(14) e X em m(12) e m(8):** A /D
- **Grupo de 1s em m(5):** /A B /C

A expressão simplificada é:

F(A,B,C,D)= (/A /B C) + (A /D) + (/A B /C)

1.b) Simplificação da função para Produto de Somas

Para simplificar a expressão em produto de somas, duplica-se o mapa de Karnaugh e segue-se o processo de simplificação, laçando-se os 0´s. Observam-se os laços em termos de produtos de somas. A expressão simplificada em produto de somas através do Mapa de Karnaugh:

AB\CD	00	01	11	10
00	X	0	1	1
01	X	1	0	0
11	X	0	0	1
10	X	0	0	1

F (A,B,C,D) = (B + C) (/A + /D) (A + /B + /C)

2.a) Conversão de $F3C_h$ em base 8

O número é $F3C_h$, que está em hexadecimal. Converte-se esse número para base 8.

1. **Converter de hexadecimal para binário:**
   - F = 1111
   - 3 = 0011
   - C = 1100

   Assim, $F3C_h$ em binário é: **1111 0011 1100, ou 111 100 111 100**

2. **Converter de binário para octal (base 8):** Agrupa-se os bits em blocos de 3 bits (da direita para a esquerda):

   111 100 111 100

   Convertendo para octal:

   - 111 = 7
   - 100 = 4
   - 111 = 7
   - 100 = 4

   Logo, $F3C_h$ em base 8 é 7474_8.

2.b) Conversão de 732_8 para base 10

Para converter o número 732_8 para base 10, utilizamos a fórmula de conversão de bases:

$$732_8=7\cdot8^2 + 3\cdot8^1 +2\cdot8^0$$

Calcula-se cada termo:

1. $7 \times 8^2 =7 \times 64 = 448$
2. $3\cdot \times 8^1 =3 \times 8 = 24$
3. $2\cdot \times 8^0 =2 \times 1 = 2$

Somam-se os resultados:

$448+24+2=474$

O número 732_8 em base 10 é 474_{10}.

3.a) Representação de -120 em binário com 8 bits (complemento de 2)

Para representar o número -120 em binário com 8 bits usando o complemento de 2:

1. **Representação de 120 em binário**: Converte-se o número positivo 120 para binário:

   $120=1111000_2$

   Como se usa 8 bits, adiciona-se um zero à esquerda:

   $120=01111000_2$

2. **Inverter os bits** (complemento de 1): Inverte-se todos os bits da representação binária:

   $01111000_2 \rightarrow 10000111_2$

3. **Somar 1 ao complemento de 1**: Adicionar 1 ao número obtido:

   $10000111_2 + 1 = 10001000_2$

A representação de -120 em binário com 8 bits (complemento de 2) é 10001000_2

3.b) Represente o número $0,9375_{10}$ na base 20. Apresente os cálculos.

Para converter o número decimal 0,9375 para a base 20, seguimos um processo similar ao de conversão para outras bases.

Passo 1: Multiplicar a parte decimal por 20

A parte decimal do número é 0,9375. Vamos multiplicar esse valor por 20 e observar a parte inteira e a parte decimal do resultado.

0,9375 × 20 = 18,75

A parte inteira é 18, e a parte decimal é 0,75.

Passo 2: Obter o primeiro dígito na base 20

A parte inteira 18 corresponde ao primeiro dígito na base 20. Para representar esse número na base 20, usamos os seguintes símbolos:

- 0 a 9 representam os mesmos valores que na base 10.
- A representa 10,
- B representa 11,
- C representa 12,
- D representa 13,
- E representa 14,
- F representa 15,
- G representa 16,
- H representa 17,
- I representa 18.

Portanto, o primeiro dígito é I.

Passo 3: Continuar com a parte decimal

Com a parte decimal 0,75 repetimos o processo:

0,75 × 20 = 15

A parte inteira agora é 15, e a parte decimal é 0. O número 15 na base 20 é representado pela letra F.

Passo 4: Resultado final

Como a parte decimal agora é zero, paramos a conversão. A representação de 0,9375 na base 20 é:

$0{,}9375_{10}=0{,}IF_{20}$

Grupo II (5 valores)

Dada a função lógica:

$$F(A,B,C,D) = \left(A + \overline{B}\right)\left(\overline{A} + C\right) + \overline{\overline{A}B} + B\overline{C}$$

4. Simplificação algébrica da função

F(A,B,C,D) = (A + /B) (/A+C) + A + /B + B/C ----Lei de Morgan

= A/A + AC + /A/B + /BC + A + /B + B/C -----distributividade

= 0 + AC + /A/B + /BC + A + /B + B/C -----x./x=0

= A (C + 1) + /A/B + /BC + /B + B/C -----distributividade

=A + /A/B + /B(C + 1) + B/C ----elemento absorvente da disjunção/elemento neutro da conjunção e distributividade

= A + /A/B + /B + B/C ---- elemento absorvente da disjunção e neutro da conjunção

= A+/B + /B + /C ----x+ /xy = x+y

= A + /B + /C ----x+x=x

5. Implementação com Portas NAND

Pontos a ter em atenção:

1 - /X = /(X.X)

2 - X+Y = /(/X . /Y)

3 - X.Y = /(/(X.Y)./(X.Y))

4 - /(X+Y) = /X . /Y

F= A+/B+/C

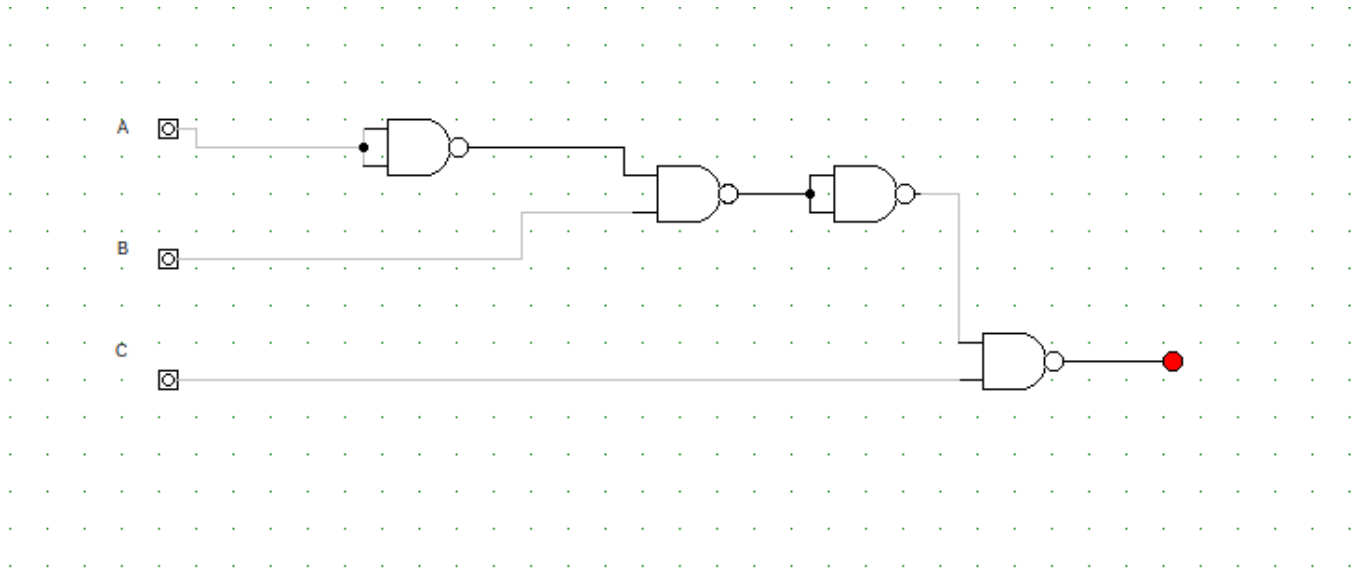
F=(A+/B) + /C

F= / (/ (A+/B) . C), 2

F= / ((/A.B) . C), 4

F = /(/(/(A.B) . /(/A.B)) . C), 3

F= /(/(/(/(A.A) . B) . /(/(A.A).B)) . C), 1



6. Implementação com Portas NOR

Pontos a ter em conta:

X + Y = /(/(X + Y) + /(X + Y)), 1

/X = /(X+X), 2

F= A + /B + /C

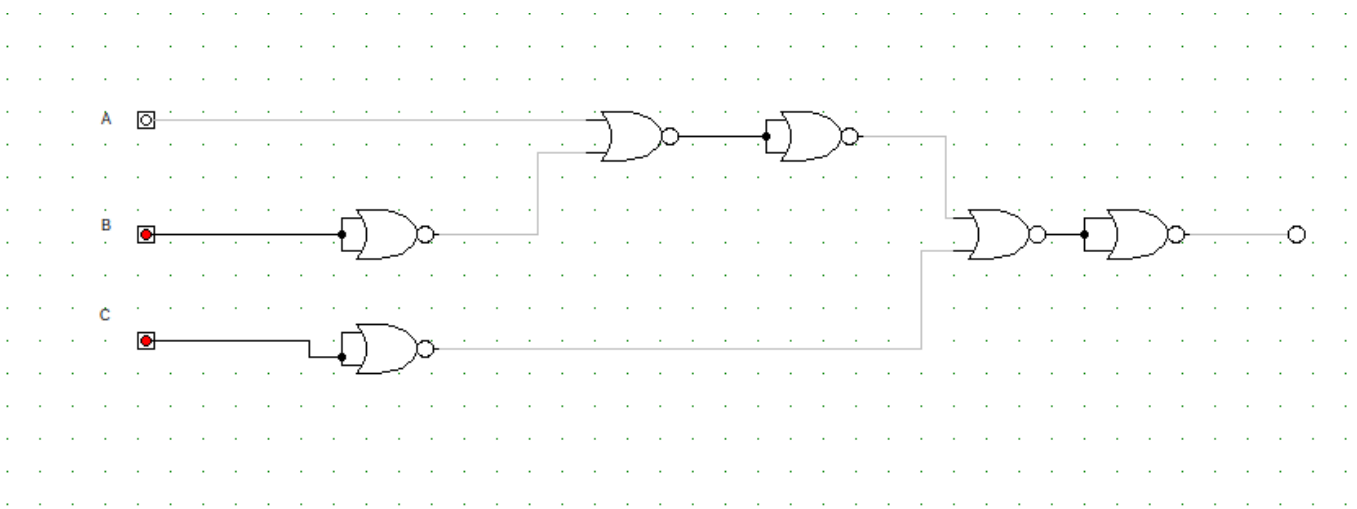
F= (A + /B) + /C

F= / (/(A + /B) + /(A + /B))+ /C, 1

F= / (/ (/(A + /B) + /(A + /B))+ /C),

F= /(/ (/ (/(A + /B) + /(A + /B))+ /C) + / (/ (/(A + /B) + /(A + /B))+ /C)) , 1

F= /(/ (/ (/(A + (/B + B)) + /(A + /(B+B)))+ /(C+C)) + / (/ (/(A + /(B+B)) + /(A + /(B+B)))+ /(C+C))) , 2



7. Implementação com Multiplexer

Para implementar F(A, B, C) usando um **multiplexer de 2 variáveis de seleção**:

1. **Escolha das variáveis de seleção:**
   - S_1=A, S_0 = B.

2. **Tabela verdade para F(A, B, C):**

A	B	F(C)
0	0	1
0	1	/C
1	0	1
1	1	1

3. **Entradas do Multiplexer (I_0, I_1, I_2, I_3):**

- I_0 = 1 (caso A=0, B=0)
- I_1 = /C (caso A=0, B=1)
- I_2 = 1 (caso A=1, B=0)
- I_3 = 1 (caso A=1, B=1)

4. **Configuração do Multiplexer:**
   Conectar A e B às entradas de seleção S_1 e S_0, respetivamente, e conectar os valores I_0, I_1, I_2, I_3 às entradas de dados do multiplexador.

Expressão final baseada no multiplexer:

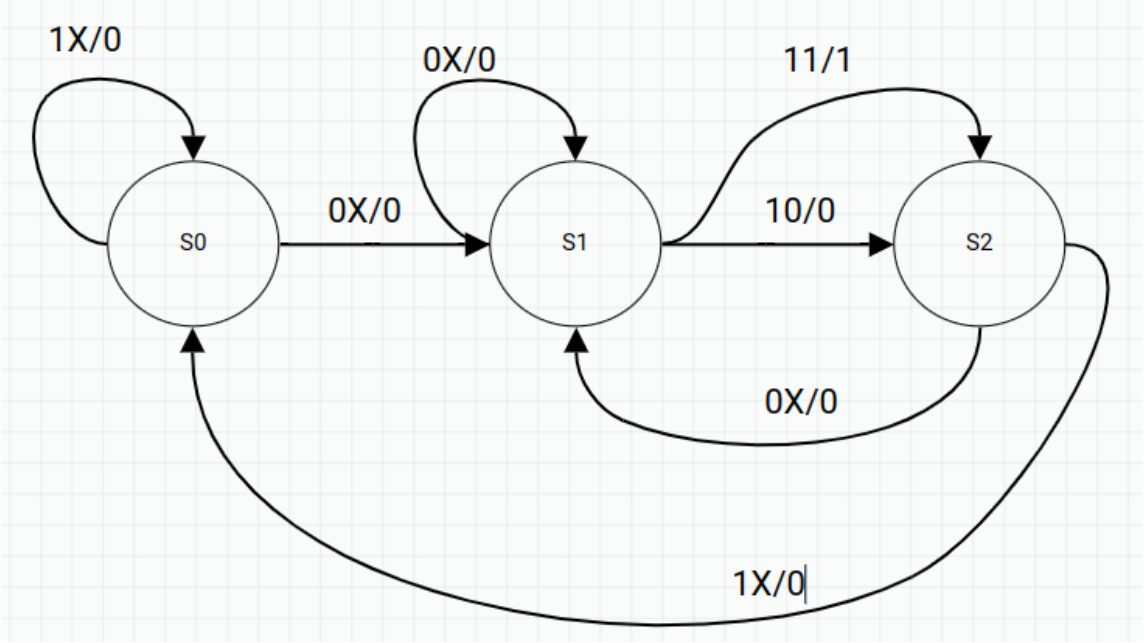
F(A,B,C) = MUX (A,B, I0=1, I1=/C, I2=1, I3=1)

8. Diagrama de Estados

1. **Definição dos estados**:
   - S1: Primeiro 1 detetado
   - S2: 0 detetado
   - S3: Sequência "101" detetada.

2. **Transições de estados**:
   - De S1:
     - Entrada 1=1: Permanece em S0
     - Entrada 1=0: Transita para S1
   - De S1:
     - Entrada 1=0: Permanece em S1
     - Entrada 1=1: Transita para S2
   - De S2:
     - Entrada 1=1 : Transita para S0
     - Entrada 1=0 : Transita para S1

3. **Saída Z**:
   - Z = Entrada 2 somente no estado S2
   - Z = 0 nos demais estados



Mapa de atribuição de estados

X1 / X0	0	1
0	S0	X
1	S1	S2

9. Tabela de transição de estados

	Atual		Entradas		Seguinte			Saída
	X1	**X0**	**E1**	**E2**	**X1**	**X0**		**Z**
S0	0	0	1	X	0	0	S0	0
S0	0	0	0	X	0	1	S1	0
S1	0	1	0	X	0	1	S1	0
S1	0	1	1	0	1	1	S2	0
S1	0	1	1	1	1	1	S2	1
S2	1	1	0	X	0	1	S1	0
S2	1	1	1	X	0	0	S0	0

D0

X1 / X0	0	1
0	/E1	X
1	1	/E1

D1

X1 / X0	0	1
0	0	X
1	E1	0

D0 = /E1 /X1 + /E1 X0

D1 = E1 X0 /X1

10. Simplificação da variável de saída

A saída Z depende do estado S_3 e da Entrada 2:

- Z = S_3·Entrada 2.

  Z = X0 X1 E2

Portanto, Z é ativada apenas quando o sistema está no estado S2 e o seu valor é igual ao da Entrada 2.

11.a) **[0,5]** Colocar no octeto menos significativo de R2 o octeto menos significativo de R2

MVBL R2, R1

MVBL
Formato: MVBL op1, op2 Flags: Nenhuma
Acão: op1 (op1 ^ FF00h) _ (op2 ^ 00FFh), copia o octecto de menor peso de op2 para o octecto de menor peso de op1.

Em alternativa, pode ser usada esta solução, que será mais complexa e provavelmente teve ajuda

MOV R1, R2; Copia o conteúdo de R2 para R1

SHR R1, 8; Desloca R1 8 bits para a direita (para remover o octeto menos significativo)

SHL R1, 8; Desloca R1 8 bits para a esquerda, colocando o octeto mais significativo de R2 em R1

11.b) **[0,5]** Colocar na posição de memória em "M" o conteúdo de R1

MOV [M], R1 **;** Armazena o conteúdo de R1 na posição de memória M

MOV
Formato: MOV op1, op2 Flags: Nenhuma
Acão: op1 op2, copia o conteúdo de op2 para op1.
Para além dos modos de endereçamento comuns a todas as instruções (conforme Secção 3.4), esta instrução permite ler e escrever no registo apontador da pilha SP, mas apenas em conjunção com o modo de endereçamento por registo: MOV SP, Rx e MOV Rx, SP. A primeira destas instruções será necessária no início de todos os programas que utilizem a pilha.

11.c)[0,5] Chamada condicional à subrotina "escrever", se a última operação aritmética/lógica teve resultado zero

CALL.Z escrever

CALL.*cond*

Formato: `CALL.cond <endereço>` *Flags*: Nenhuma

Acção: chamada condicional a uma subrotina baseado no valor de um dado bit de estado.
As versões disponíveis são:

Condição	Transporte	Sinal	Excesso	Zero	Interrupção	Positivo
Verdade	CALL.C	CALL.N	CALL.O	CALL.Z	CALL.I	CALL.P
Falso	CALL.NC	CALL.NN	CALL.NO	CALL.NZ	CALL.NI	CALL.NP

Caso a condição se verifique, comporta-se como uma instrução CALL. Caso contrário, funciona como um NOP. Normalmente <endereço> é especificado com uma etiqueta.

1. d) [0,5] Coloca na pilha o conteúdo de R2

PUSH R1

PUSH

Formato: `PUSH op` *Flags*: Nenhuma

Acção: *M[SP]* ← *op*, *SP* ← *SP - 1*, coloca `op` no topo da pilha.

12. [3] Elabore um programa no assembly do P3. O programa recebe uma frase em R1 com um conjunto de letras. O programa deve contar todas as vogais minúsculas sem acentos que encontra na frase e apresentar o resultado hexadecimal da contagem em R3.

Exemplo:

R1 = 'Arquitetura de Computadores'

R3 = 11

Pretende-se que conte as vogais a verde em: 'Arq**ui**t**e**t**u**r**a** d**e** C**o**mp**u**t**a**d**o**r**e**s'

Este caso de teste ficaria com R3 = 11

```
        ORIG 8000h          ; Endereço da string
TEXTO   STR ' Arquitetura de Computadores ',0 ; String de exemplo (pode ser alterada) com caractere nulo para finalizar a string
        ORIG 0000h          ; Endereço inicial do programa
START:  MOV R3,R0           ; R3 será o contador de vogais, inicializa com 0
        MOV R1, TEXTO       ; R1 aponta para o início da string
LOOP:   MOV R2, M[R1]       ; Carrega o caractere atual da string em R2
        CMP R2, 0           ; Verifica se é o fim da string (caractere nulo)
        JMP.z FIM           ; Se for o fim, salta para o final
        ; Verifica se o caractere é uma vogal minúscula
        CMP R2, 'a'
        JMP.z VOGAL
        CMP R2, 'e'
        JMP.z VOGAL
        CMP R2, 'i'
        JMP.z VOGAL
        CMP R2, 'o'
        JMP.z VOGAL
```

```
        CMP R2, 'u'
        JMP.z VOGAL
        JMP PROXIMO         ; Se não for vogal, vai para o próximo caractere
VOGAL:  INC R3              ; Incrementa o contador de vogais
PROXIMO:INC R1              ; Avança para o próximo caractere
        JMP LOOP            ; Continua o loop
FIM:  NOP
                            ; Resultado final está em R3 (número de vogais)
                            ; Fim do programa
```